UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/628,959	07/28/2003	Eitan Hefetz	34874-020 UTIL	6174

64280 7590 09/30/2010

MINTZ, LEVIN, COHN, FERRIS, GLOVSKY & POPEO, P.C.
ONE FINANCIAL CENTER
BOSTON, MA 02111

EXAMINER	
PATEL, MANGLESH M	

ART UNIT	PAPER NUMBER
2178	

MAIL DATE	DELIVERY MODE
09/30/2010	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 10/628,959
Filing Date: July 28, 2003
Appellant(s): HEFETZ ET AL.

James P. Cleary
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 7/6/2010 appealing from the Office action mailed 1/5/2010.

(1) Real Party in Interest

The examiner has no comment on the statement, or lack of statement, identifying by name the real party in interest in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The following is a list of claims that are rejected and pending in the application: Claims 1-9, 18-20 and 26.

(4) Status of Amendments After Final

The examiner has no comment on the appellant's statement of the status of amendments after final rejection contained in the brief.

(5) Summary of Claimed Subject Matter

The examiner has no comment on the summary of claimed subject matter contained in the brief.

(6) Grounds of Rejection to be Reviewed on Appeal

The examiner has no comment on the appellant's statement of the grounds of rejection to be reviewed on appeal. Every ground of rejection set forth in the Office action from which the appeal is taken (as modified by any advisory actions) is being maintained by

the examiner except for the grounds of rejection (if any) listed under the subheading "WITHDRAWN REJECTIONS." New grounds of rejection (if any) are provided under the subheading "NEW GROUNDS OF REJECTION."

(7) Claims Appendix

The examiner has no comment on the copy of the appealed claims contained in the Appendix to the appellant's brief.

(8) Evidence Relied Upon

7,316,003	Dulepet	1-2008

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claims 1-9, 18-20 and 26 remain rejected under 35 U.S.C. 102(e) as being anticipated by Dulepet (U.S. 7,316,003, filed Dec 18, 2002).

Regarding Independent claim 1, a computer-implemented method comprising:

- Providing a design-time translator and a run-time translator that both correspond to a defined page element. The run-time translator and design time translator configured on a processor;
- During design-time for a page, invoking the design-time translator for a page template including the defined page element having content components, said design-time invoking resulting in the defined page element in the page template being translated into a design-time representation of the content components in the page , the design-time representation being rendered in accordance with a predefined layout of a

container for the content components, the page template being available to a plurality of remote users of a portal, the content components including a first content component and a second content component, the first content component configured as static content with a run-time behavior determinable at design-time, and the second component configured as dynamic content with a run-time behavior not determinable at run-time, such that at design-time a tag is used to represent the dynamic content on the page rendered at design-time; and

- During run-time for the page, invoking the run-time translator for the page template, said run-time invoking resulting in the content components being obtained and the defined page element in the page template being translated into a run-time presentation of the obtained one or more content components in accordance with the layout of the container, wherein the second component configured as dynamic content is determined and obtained in parallel, at run-time using threading, with other dynamic content stored in blocks without ordering in a content storage medium to render the dynamic content of the second component rather than the tag used during design-time.

Dulepet teaches creation/editing of a dynamic web page using a WYSIWYG editor. He describes the use of a design time engine which in response to a controller-deployed dynamic page request, the design time engine replaces the dynamic code JSP elements

with a design time component, such a component comprises a content placeholder representative of content that would have been generated by a JSP container if the controller had deployed the dynamic code element to an executing JSP container. Furthermore the page template is available to a plurality of remote users of a portal because the Merged model in fig 2 synchronizes the updated from the editor in design-time to the application database server thus making it available to the remote users of the portal. He then describes that during run-time upon receiving the source code, JSP container replaces dynamic source code elements within the source code with dynamically generated page content, and returns a dynamically generated web page (see abstract, fig 2-3, column 1, lines 5-67, column 2, lines 10-50, column 2, lines 55-67, column 3, lines 1-52, column 6, lines 5-58 & column 5, lines 50-67). **Furthermore the dynamic component is determined and obtained in parallel at run-time using threading because Dulepet discloses that controller 230 may deploy web page source code/or java servlets to multiple JSP containers. Dulepet (see column 6, lines 55-67) provides a suggestion that controller 230 handles 2 or more concurrently running tasks for processing multiple JSP containers thereby resulting in rendering of the dynamic content components.**

Regarding Dependent claim 2, which depends on claim 1, Dulepet discloses wherein said invoking the design-time translator further results in presentation of a WYSIWYG layout editor using the design-time representation of the one or more content components

in the page (column 2, lines 10-50, column 2, lines 55-67, including the explanation provided in the Independent claims).

Regarding Dependent claim 3, which depends on claim 2, Dulepet discloses wherein the said invoking the design-time translator further results in client-side scripting components being included in the design-time representation to form at least part of the WYSIWYG layout editor and enable adding a content component to a content container using a drag-and-drop action (column 2, lines 10-50, column 2, lines 55-67, including the explanation provided in the Independent claims).

Regarding Dependent claim 4, which depends on claim 2, Dulepet discloses wherein the page template comprises a portal page template, and the WYSIWYG layout editor comprises a WYSIWYG portal page layout editor (column 6, lines 5-58 & column 5, lines 50-67, including the explanation provided in the Independent claims).

Regarding Dependent claim 5, which depends on claim 4, Dulepet discloses wherein the defined page element comprises a custom Java Server Page tag and the design-time translator and the run-time translator comprise Java Server Page tag handlers for the custom Java Server Page tag, and wherein the run-time translator obtains portal dynamic content according to the portal page template and the design-time translator does not (column 1, lines 5-67, column 2, lines 10-50, column 2, lines 55-67, column 3, lines 1-52, including the explanation provided in the Independent claims).

Regarding Independent claims 6 and 18, an article comprising a machine-readable storage medium storing instructions operable to cause one or more machines to perform operations comprising: Providing a design-time translator and a run-time translator that both correspond to a defined page element, the run-time translator and design time translator configured on a processor; during design-time for a page, invoking the design-time translator for a page template including the defined page element having-content components, said design-time invoking resulting in the defined page element in the page template being translated into a design-time representation of the content components in the page, the design-time representation being rendered in accordance with a predefined layout of a container for the content components, the page template being available to a plurality of remote users of a portal, the content components including a first content component and a second content component, the first content component configured as static content with a run-time behavior determinable at design-time, and the second component configured as dynamic content with a run-time behavior not determinable at run-time, such that at design-time a tag is used to represent the dynamic content on the page rendered at design-time; and during run-time for the page, invoking the run-time translator for the page template, said run-time invoking resulting in the content components being obtained and the defined page element in the page template being translated into a run-time presentation of the obtained one or more content components in accordance with the layout of the container, wherein the second component configured as dynamic content is determined and obtained in parallel, at run-time using threading, with

other dynamic content stored in blocks without ordering in a content storage medium to render the dynamic content of the second component rather than the tag used during design-time.

Dulepet teaches creation/editing of a dynamic web page using a WYSIWYG editor. He describes the use of a design time engine which in response to a controller-deployed dynamic page request, the design time engine replaces the dynamic code JSP elements with a design time component, such a component comprises a content placeholder representative of content that would have been generated by a JSP container if the controller had deployed the dynamic code element to an executing JSP container. Furthermore the page template is available to a plurality of remote users of a portal because the Merged model in fig 2 synchronizes the updated from the editor in design-time to the application database server thus making it available to the remote users of the portal. He then describes that during run-time upon receiving the source code, JSP container replaces dynamic source code elements within the source code with dynamically generated page content, and returns a dynamically generated web page (see abstract, fig 2-3, column 1, lines 5-67, column 2, lines 10-50, column 2, lines 55-67, column 3, lines 1-52, column 6, lines 5-58 & column 5, lines 50-67). **Furthermore the dynamic component is determined and obtained in parallel at run-time using threading because Dulepet discloses that controller 230 may deploy web page source code/or java servlets to multiple JSP containers. Dulepet (see column 6, lines 55-67) provides a suggestion that controller 230 handles 2 or more concurrently running**

tasks for processing multiple JSP containers thereby resulting in rendering of the dynamic content components.

Regarding Dependent claim 7, which depends on claim 6, Dulepet discloses wherein translating the placeholder during design-time comprises adding code enabling editing of the portal page, the added code forming at least part of the WYSIWYG portal layout editor (column 2, lines 10-50, column 2, lines 55-67, including the explanation provided in the Independent claims).

Regarding Dependent claim 8, which depends on claim 7, Dulepet discloses wherein the added code comprises client-side scripting that enables addition of a content component to a content container in the portal page using a drag-and-drop action (column 2, lines 10-50, column 2, lines 55-67, including the explanation provided in the Independent claims).

Regarding Dependent claim 9, which depends on claim 6, Dulepet discloses wherein the placeholder comprises a custom Java Server Page tag, said translating the placeholder during design-time comprises invoking a design-time Java Server Page tag handler corresponding to the custom Java Server Page tag, and said translating the placeholder during run-time comprises invoking a run-time Java Server Page tag handler corresponding to the custom Java Server Page tag (column 1, lines 5-67, column 2, lines

10-50, column 2, lines 55-67, column 3, lines 1-52, including the explanation provided in the Independent claims).

Regarding Dependent claim 19, which depends on claim 18, Dulepet discloses wherein the first tag handler interprets the portal page template by including client-side scripting that enables addition of a content component to a content container in the portal page template using a drag-and-drop action (column 2, lines 10-50, column 2, lines 55-67, including the explanation provided in the Independent claims)

Regarding Dependent claim 20, which depends on claim 18, the claim describes a system that contains the same limitations as claim 5 and is rejected under the same rationale.

Regarding Independent claim 26, Dulepet discloses A computer-implemented method for selectively interpreting a portal page layout template, the method comprising: providing a design-time translator and a run-time translator, the design-time translator and the run-time translator both corresponding to a same defined page element or placeholder, and being invoked based on a current mode of operation, the run-time translator and design time translator configured on a processor (See fig 4a & 4B & 4C & abstract showing a design time translator and run-time translator corresponding to a defined page element and being invoked by the pre/post processors); translating a placeholder in a portal template during design-time into a representation of a container

designed to present portal content using a single template file for both run-time and design-time, the container representation showing a layout context for the portal content that will be obtained and revealed at run-time, the container representation also directly presenting dynamic content source information for the content container (See fig 4a & 4B & 4C & abstract & column 6, lines 25-64, which shows translating a placeholder during design-time in 4A into a representation of a container using a template file as disclosed in both 4A and 4B.);presenting a WYSIWYG portal layout editor using the container representation designed to present the portal dynamic content, the WYSIWYG portal layout editor facilitating editing of the portal template and the resulting portal page (column 6, lines 1-59, wherein the WYSIWYG editor is presented using the container representation for editing of the portal template and portal page); obtaining portal dynamic content during run-time from a dynamic content source, the placeholder in the portal template being translated into a presentation of the container containing the obtained portal dynamic content component, wherein the dynamic content is determined and obtained in parallel, at run-time using threading with other dynamic content stored in blocks without ordering in a dynamic content source providing a storage medium, to render the dynamic content rather than a tag used during design-time to represent the dynamic content (see fig 4a-4c, showing dynamic content during run-time); and parsing and locating, by a run-time application, the placeholders in the template and replacing them with the run-time content components, and at design-time, parsing and rendering the same template with a representation of the content components, in place of the actual run-time content components, to reveal the run-time layout during design of the template

(see fig 4a-4c & column 6, lines 1-67, disclosing placeholders in the template being replaced with the run-time content, including design-time previews of the dynamic content with placeholders). **Furthermore the dynamic component is determined and obtained in parallel at run-time using threading because Dulepet discloses that controller 230 may deploy web page source code/or java servlets to multiple JSP containers. Dulepet (see column 6, lines 55-67) provides a suggestion that controller 230 handles 2 or more concurrently running tasks for processing multiple JSP containers thereby resulting in rendering of the dynamic content components.**

It is noted that any citation [[s]] to specific, pages, columns, lines, or figures in the prior art references and any interpretation of the references should not be considered to be limiting in any way. A reference is relevant for all it contains and may be relied upon for all that it would have reasonably suggested to one having ordinary skill in the art. [[See, MPEP 2123]]

(10) Response to Argument

(a) Appellant Argues:

However, Dulepet fails to teach or suggest a runtime translator that corresponds to the same defined page element as a design-time translator. This feature of the claimed invention provides a template-based page layout capability such that the layout can execute both at runtime and design-time. On the contrary, Dulepet teaches only an editor that develops a merged model representation of a dynamic web page, merging static and dynamic content in an HTML page that ostensibly can only be executed in

a general runtime environment. Accordingly, Dulepet cannot be reasonably argued to disclose the limitation of both a design time translator and a runtime translator that both correspond to a defined page element., (pg 14, paragraph 4- pg 15, paragraph 1)

Both the Appellant and the Examiner acknowledge that JSP scripting elements are executed in a runtime environment as suggested by Dulepet. Dulepet states that "the visual editor view may display a visual representation of the dynamic content" (column 3, lines 3-45) this is done using JSP scripting elements or tags that represent a container for dynamic content thereby suggesting a design-time representation of markup content component. He states in column 3, lines 8-11, "**The visual editor view may display a visual representation of the dynamic web page that is essentially similar to a visual representation of the dynamic web page, as it would be displayed in a typical web browser**." Dulepet herein describes a visual editor and design time translation of dynamic code to a defined page element, he then teaches that it is similar to what would be displayed at Runtime for the defined page element, that is when displayed in a typical web browser. Thus Dulepet therefore teaches a design time translator and a runtime translator that both correspond to a defined page element.

Appellants specification discloses that placeholders are used in Design-time only for displaying the layout of the document but not actually obtaining dynamic content for display which instead is only done at run-time, Dulepet also teaches the use of placeholders/ JSP containers during Design-time (see column 6, lines 5-65).

Appellants specification discloses:

"Thus, the portal designer can essentially see the actual structure and appearance of a page at design-time along with the other elements (e.g. HTML tags and components, image, etc) that are placed in a template, except for the run-time content itself..." paragraph 7

"In yet another aspect, a portal system includes a WYSIWYG portal layout editor that uses a selectively interpreted portal page template to reveal a WYSIWYG layout content for dynamic content without obtaining the portal dynamic content." Paragraph 13

"The template can provide WYSIWYG layout for portal pages at design-time and can ensure that the user sees the same representation of a page both at design-time and run-time, with the only difference being that dynamic content is represented by a content container and not actually presented at design-time." Paragraph 30

"At design-time, the presentation corresponds directly to what will be presented at run-time, with the exception of the actual run-time content being withheld." Paragraph 39

"At run-time, the placeholders are replaced with the actual components presenting their content, while at design-time the place holders present content components' respective representations to simulate the run-time presentation." Paragraph 46

(b) Appellant Argues:

The office assets that Dulepet teaches obtaining multiple content items to multiple containers, and that concurrently running two or more tasks defines the term "threading". But, these alleged teachings Dulepet, while relevant to modern operating systems, do not suggest the limitations of using threading to determine and obtain dynamic content in parallel with other dynamic content stored in blocks without ordering. In fact, Dulepet does not even mention the term "parallel" or any term similar. Fig. 5 in Dulepet clearly shows a recursive method for determining whether dynamic source code is present in the web page source code after the source code has been parsed.. (pg 15, paragraph 3, & pg 16, paragraph 1)

The Examiner Respectfully Disagrees: Applicants specification describes at paragraph 57 "The dynamic run-time content can be gathered in parallel (e.g. by using **java multithreading**) to improve performance". Although

the claims don't explicitly disclose multi-threading, Dulepet does teach use of Java and JSP which already support multithreading or threading in the underlying Java virtual Machine.

Dulepet teaches obtaining multiple content items to multiple JSP containers (see column 6, lines 60-65). Threading is concurrently running two or more tasks, and is common in most if not all modern operating systems. Dulepet teaches obtaining multiple content items which is done during parsing or runtime so that the server fetches the dynamic content and then displaying them in the users computer obviously not one at a time but all together for display thereby including threading. Furthermore the dynamic content is stored in blocks without ordering because the actual underlying dynamic content has to be retrieved from a repository and does not reside in the underlying document to have any specific ordering.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejection should be sustained.

Respectfully Submitted,

/Manglesh Patel/
AU 2178
9/23/2010

Conferees:

/Stephen S. Hong/
Supervisory Patent Examiner, Art Unit 2178

Stephen S. Hong
Supervisory Patent Examiner
Art Unit 2178

/William L. Bashore/
Supervisory Patent Examiner, Art Unit 2175